CSS Modules

1. *Localized* class names *to avoid global conflicts*
2. *Styles* are scoped *to individual components*
3. *Helps* in creating *component specific* styles
4. *Automatically* generates *unique class* names
5. *Promotes modular* and *maintainable* CSS
6. *Can use* alongside *global CSS* when needed

Let's learn how can we use **CSS module** in our project

So for better understanding we are going to add some **CSS** in **Healthy Food Project** that we made previously

So Open that project and start adding **CSS**

add traditional CSS inside `App.jsx`

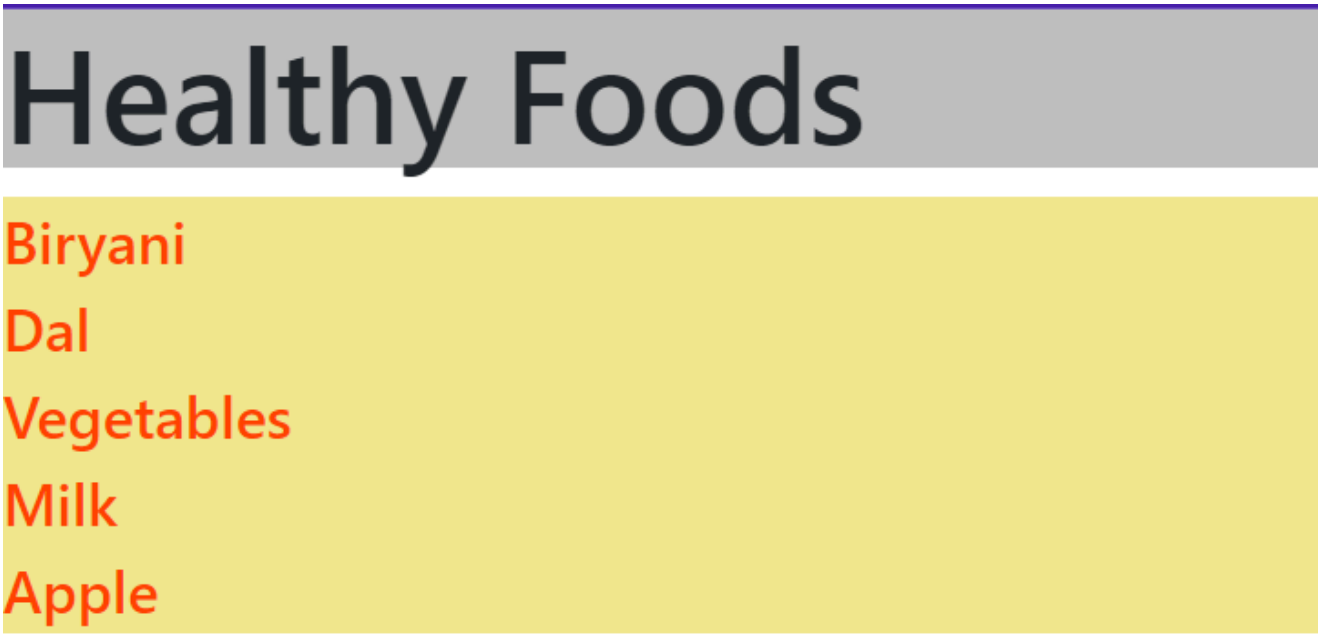
```
.kg-item {
  background-color: khaki;
}

.kg-span {
  font-weight: 500;
  color: orangered;
```

```
}

.food-heading {
  background-color: silver;
  color: brown;
}
```

Now the page will look like this



So the problem in this is we will be exhausted if the project starts scaling up so at that point it will become very hefty and we can never be able to maintain

So for that reason we are going to use **CSS Module**

for modularity purpose

Let's create a file `Item.module.css` inside `src` folder

do these changes

```
.kg-item {
  background-color: khaki;
```

```
}

.kg-span {
  font-weight: 500;
  color: orangered;
}
```

Now we have to **import** it on the file where it is being used i.e. `Item.jsx` and do these changes else **CSS** will not work

```
import styles from "./Item.module.css";

const Item = ({ foodItem }) => {
  return (
    <li className={`${styles["kg-item"]}`}>
      <span className={styles["kg-span"]}>{foodItem}
</span>
    </li>
  );
};

export default Item;
```

`import styles from "./Item.module.css";`

```
<li className={`${styles["kg-item"]}`}> <span
className={styles["kg-span"]}>{foodItem}</span></li>
```

these are the changes we made to apply the **CSS** modular

Now the page will look like this

Healthy Foods

Biryani

Dal

Vegetables

Milk

Apple